# Efficient Procedure Improving Precision of High Conditioned Matrices in Electronic Circuits Analysis

*David CERNY, Josef DOBES, Stanislav BANAS*

Dept. of Radioelectronics, Czech Technical University in Prague, Technická 2, 166 27 Praha 6, Czech Republic

{cernyd1, dobes, banassta}@fel.cvut.cz

Submitted September 22, 2017 / Accepted August 30, 2018

**Abstract.** *In this article, we propose several improvements that could be done to SPICE simulator. The first is based on a functional implementation of device models. The advantages of functional implementation are demonstrated on basic Shichman-Hodges model of MOS transistor. It starts with a description of primary algorithms used in SPICE simulator for the solution of circuits with nonlinear devices and identify the problems that can occur during simulations. Main part of the article is devoted to improved factorization procedure for simulation of the nonlinear electronic circuits. The primary intention of the proposed method is to increase final precision of the result in a case of high condition linear systems. The procedure is based on a use of the iterative methods for solution of nonlinear and linear equations. Utilizing those methods for one iterative process helps to reduce memory consumption during simulation computation, and it can significantly improve simulation precision. The procedure allows to use enumeration with definable precision in a very efficient way.*

## Keywords

SPICE simulator, functional programming, Lisp, iterative methods, factorization procedures, conditionality of linear systems, simulation precision

## 1. Introduction

Recent development of the simulation of electronic circuits can be divided into several directions. The first is focused mainly on improving device models of the program SPICE. They go from enhancing possibilities given by standard models, through implementing entirely new ones to constructing new macro-models formed by standard models. From the recent publications, we can point out the following articles [1–9] that are entirely or partly focused on model improvement in the SPICE simulators. The second direction of the development is targeted to algorithms used during simulation. It can be divided into the two subsections. The first subsection is dedicated to concrete simulation type or simulation problem where standard simulation core could not be used because given device models or simulation problem is not applicable to SPICE computation procedure. The most often it is the simulation of various DC-DC circuits, but many other articles have been published for enhancing core to simulate magnetic features or another circuit behavior and effects. From recent works focused on strengthening simulation core we can mention [10–17]. It could be seen this is very active field of development. The second subsection is focused to SPICE core algorithm performance or performance of an entire simulation process. SPICE core wasn't changed much since it was first developed; therefore it is interesting to adapt it to present hardware or software, mainly with an intention of improving simulator performance or accuracy of the results. From recent publications in this area we recommend [18–20]. Moreover, new (and often unusual) modes of analysis and optimization have recently been implemented to the SPICE-family programs that are *very demanding* to the precision of the algorithms [21–27]. This article belongs to the last mentioned subsection with one important note that it does try to enhance standard simulation procedure but it completely redefine the process with the new one where standard procedure is replaced with recursive and factorization free evaluation algorithms and as suitable programming tool is used functional language Lisp. It can reduce memory requirements, computation time and add a new factor of variability not only during device modeling but also during specification of the simulation. The idea originates from adaptive environment for solving of extra large datasets and combines advantages from today's use of functional and aspect-oriented languages [28–31].

## 2. Models and Virtual Representation

The real electronic devices such as resistors, capacitors, diodes, or MOS transistors are represented in the simulation programs by its software models. Those models are interpreted in the simulation core as a set of mathematical equations with the purpose of accurately approximating device behavior in a particular region of operation. It is a fact that device models become not very reliable when simulation gets out of assumed operation bounds. It is mainly caused by the developers' intent to keep models equations simple and efficient with fast convergence behavior while they are still precise within their operation point bounds [32], [33]. If we

assume that simulation takes place within the device's model bounds, and there is no error in circuit design the next stage of device modeling is a rate of affection by simulation changes. Device models are usually defined as closed structures that enter a simulation by calling its extern functions. This leads to a situation where models are only particular part of the simulation and can be affected only as much as arguments of their functions allow. They can be enhanced when we stop taking model devices as closed structures that can return only numeric values and rewrite them in a functional form.

# 3. Functional Device Models

In order to demonstrate the functional device modeling (FDM), let us start with the very simple (Shichman-Hodges) model of MOS transistor used in SPICE as described in [34]. The drain-source current $I_{DS}$ of an n-channel device is then

$$I_{DS} = \begin{cases} 0 & \text{for } V_{GS} \leqq V_{TH}, \\ \beta_f V_{ON}^2 X_\lambda & \text{for } 0 < V_{ON} \leqq V_{DS}, \\ \beta_f V_{DS}(2V_{ON} - V_{DS}) X_\lambda & \text{for } 0 < V_{DS} < V_{ON} \end{cases} \quad (1)$$

where each case defines different region of an operation cut-off, saturation, and linear. Variables $V_{ON}$ and $X_\lambda$ are substitutions defined as $V_{ON} = V_{GS} - V_{TH}$ and $X_\lambda = 1 + \lambda V_{DS}$. Parameter $\beta_f$ (for a forward transconductance) is substitution

$$\beta_f = \frac{K}{2}\frac{W}{L_{EFF}} \quad (2)$$

where $L_{EFF} = L - 2L_D$ is effective channel length corrected for the lateral diffusion, $L_D$, of the drain and source, and

$$V_{TH} = V_{TH0} + \gamma\left(\sqrt{\phi - V_{BS}} - \sqrt{\phi}\right) \quad (3)$$

is the threshold voltage in the presence of a bulk-gate bias, $V_{BS} < 0$. Parameters $V_{TH0}$, $K$, $\gamma$, $\phi$ and $\lambda$ are the electric parameters of the MOSFET model, representing the zero-bias threshold voltage, transconductance factor, bulk threshold parameter, surface potential, and output conductance factor in saturation, respectively. In our simulations, we are going to use the n-channel devices only with bulk connected to source. Therefore, we can simplify the model putting $V_{TH}$ to a constant value.

The variables $V_{DS}$, $V_{GS}$ depend on circuit definition and current state. At the time of model definition, those variables remain unknown and will be evaluable at the moment of circuit simulation. Also, from the point of view of the simulation core, it is not clear how many unknown variables each model introduce to a circuit. One possible solution is to a make generic get/set function returning values by their reference. Better implementation can be done utilizing a two or more stage self-redefinition mechanism denoted as "functional chaining". In the first stage model functions returns, as a result, another function with internal variables adapted to simulation unknowns as voltages, currents or temperature. In the second stage through the chaining procedure, all unknown variables will evaluate itself according to simulation state.

Functions defined through this mechanism can be treated as values despite that they are unknown. That allows computation core to save execution time and handle those functions, referred as functionals, as any normal number and store them in vectors, matrices or sparse matrix systems. As a demonstration of the previous statement, a complete definition of previously mentioned equations for a simple model of the MOS transistor in functional language Lisp follows:

```
(defmethod mos_current_drain
  ((mos class-mos) vg vs vd)
  #'(lambda ()
    (let*
      ((v_gs (- (eval vg) (eval vs) ))
       (v_ds (- (eval vd) (eval vs)))
       (v_on (- v_gs (VTH mos)))
       (x_lambda
         (+ 1 (* (LAMBDA mos) v_ds))))
      (cond
        ((< vgs (VTH mos)) 0)
        ((and (< 0 v_ds) (< v_ds v_on))
          (*
            (BETA mos) v_ds x_lambda
            (- (* 2 v_on) v_ds)))
        ((and
          (< 0 v_on )
          (< v_on v_ds))
          (* (BETA mos) x_lambda
            (expt v_on 2)))))))
```

The method denoted as "mos_current_drain" is a wrapper (a symbolic representation) for the inner functional definition (see a more detailed description of functionals in [35]):

```
(defmethod mos_current_drain
  ((mos class-mos) vg vs vd)
  #'(lambda ()
```

Specifically, it is defined without any parameters, but the wrapper function, which, on its call, will set up all internal variables of the device model to references passed through the input argument list. Those arguments vg, vs, and vd are simulation variables and will be set to circuit voltages concerning a location where the device is placed in a circuit. It is done by calling "eval" function during the enumeration:

```
(vgs (- (eval vg) (eval vs)))
```

For completeness, we need to note that for each nonlinear model there have to be defined all derivatives for all nonlinear functions and time dependent parameters. They will be used for the generation of Jacobian matrix that is required by Newton-Raphson algorithm. In our case, it has to be the two partial derivatives that will result in additional functions wrapping their functionals:

```
(defmethod mos-current-drain-dvgs)
(defmethod mos-current-drain-dvds)
```

| 1M Operations | Addition | Multiplication | Division | Division by Constant |
|---|---|---|---|---|
| C Double (ms) | 3.1 | 2.8 | 24.3 | 1.9 |
| C Arbitrary (ms) | 31.9 | 69.8 | 137.0 | 139.0 |
| Lisp Double (ms) | 1126.5 | 1143.3 | 1127.4 | 1103.2 |
| Lisp Rationals (ms) | 1778.3 | 1431.3 | 1514.0 | 1471.5 |

**Tab. 1.** Differences between run times of one million operations in C and Lisp languages.

The full definition requires derivation for each nodal voltage direction. It will result in growing of Jacobian matrix during evaluation. More information about the concepts of functional programming can be found in [36]. Comprehensive definition of nonlinear device modeling in Lisp with explanation of further steps can be found in [37] and [38].

It should be noted that functional language can be very beneficial from a point of view of simulation variability and definition. It can improve experience of the users during simulation, device model definition and creation. On the other hand it will be very inappropriate to use it for demanding numerical operations that are required by simulation analysis. It is still very difficult to get any closer with any other languages to performance of optimized code written in Fortran or C language. In Tab. 1 (it is expanded version of [35]), there is a comparison between performance of basic operations in the C and Lisp programming languages.

# 4. Device Models and Simulation

For each simulation type, there is a need to perform certain steps to obtain correct results. Basic DC simulation includes solving a linear system compiled by modified nodal formulation (MNF). It is a fundamental part of any other type of simulation and the essential starting point of transient analysis. Transient simulation characterizes device behavior, mostly nonlinear, with respect to time. It includes solving sets of nonlinear equations. The procedure that is commonly used is presented in the Algorithm 1.

**Algorithm 1.** Standard Simulation Procedure

```
for TIMELINE do
   INITIAL ENUMERATION
   LINEAR SYSTEM (LU Factorization)
   repeat {NONLINEAR SYSTEM (Newton-Raphson)}
      JACOBIAN MATRIX (LU Factorization)
      NEXT ESTIMATE (vector-matrix product)
      RESIDUAL (vector norm)
   until STOPPING CRITERIA
   if not (CONVERGENCE) then
      return CONVERGENCE PROBLEM
   end if
end for
```

From the point of view of the memory handling and computation stability, it is very fast and efficient procedure. The resulting sparse matrix after LU factorization introduces very small amount of new fill-ins, and accompanied with forward elimination & back substitution, it produces result in very short time. However, there are some special cases when usage of supplementary algorithm can be required to improve stability of simulation or precision of the result. For instance, settings of the circuit devices can significantly increase condition number of matrix produced by MNF. It can introduce to already complicated problem additional complexity, low density of sparse matrix together with high matrix condition number can cause numerical errors during floating point operations followed by substantial decrease in final precision.

# 5. LU Factorization

The LU factorization (LUF) is an efficient algorithm able to find a solution of linear system defined by highly sparse matrices. It was implemented into SPICE and other simulation programs [39]. It can be briefly defined as

$$\mathbf{Ax} = \mathbf{b} \rightarrow \mathbf{A} = \mathbf{LU} \tag{4}$$

where $\mathbf{U}$ (upper) and $\mathbf{L}$ (lower) are two triangular submatrices that lead to a solution of a linear system as

$$\mathbf{Ax} = \mathbf{LUx} \rightarrow \mathbf{Ly} = \mathbf{b},\ \mathbf{Ux} = \mathbf{y}. \tag{5}$$

The problematic fact about the LU factorization is that it is a very demanding operation consuming much memory and computation time. Even though there have been published many articles on an optimization of LU factorization [40–42], they do not solve the main problem that we believe lies in the condition number of the factorized matrix. High condition number can affect final precision of direct factorization method. In Tab. 2, a dependency is presented of the total number of required operations needed for matrix factorization on given dimension. It additionally shows the development of the number of required iterations of the algorithm and matrix sparsity. The visual comparison is shown in Fig. 1. LUF method is usually performed by the Newton-Raphson iterative algorithm. It can be performed many times before algorithm finds a solution of a nonlinear system. It is obvious that obtaining final solution is very time demanding process and should not be repeated. Also once the solution of final LUF is done the precision of the result cannot be improved by LUF algorithm in other way that recomputing it again with higher (or even arbitrary) precision numbers.

| Dim. | Matrix Density | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **100** | **90** | **Diff 90** | **80** | **Diff 80** | **70** | **Diff 70** | **60** | **Diff 60** | **50** | **Diff 50** |
| 50 | 8.45E+04 | 8.34E+04 | 1.65E+03 | 8.28E+04 | 1.65E+03 | 8.08E+04 | 3.64E+03 | 8.08E+04 | 3.69E+03 | 8.04E+04 | 4.06E+03 |
| 100 | 7.55E+05 | 7.50E+05 | 7.96E+03 | 7.47E+05 | 7.96E+03 | 7.33E+05 | 2.28E+04 | 7.39E+05 | 1.67E+04 | 7.34E+05 | 2.12E+04 |
| 150 | 3.02E+06 | 3.00E+06 | 2.28E+04 | 2.99E+06 | 2.28E+04 | 2.94E+06 | 7.12E+04 | 2.97E+06 | 4.85E+04 | 2.95E+06 | 6.13E+04 |
| 200 | 8.36E+06 | 8.33E+06 | 5.02E+04 | 8.31E+06 | 5.02E+04 | 8.21E+06 | 1.56E+05 | 8.26E+06 | 1.05E+05 | 8.23E+06 | 1.33E+05 |
| 250 | 1.88E+07 | 1.87E+07 | 1.32E+05 | 1.87E+07 | 1.32E+05 | 1.85E+07 | 2.84E+05 | 1.86E+07 | 1.95E+05 | 1.86E+07 | 2.44E+05 |
| 300 | 3.68E+07 | 3.67E+07 | 2.74E+05 | 3.66E+07 | 2.74E+05 | 3.63E+07 | 5.20E+05 | 3.65E+07 | 3.26E+05 | 3.64E+07 | 4.07E+05 |
| 350 | 6.55E+07 | 6.53E+07 | 4.33E+05 | 6.50E+07 | 4.33E+05 | 6.47E+07 | 7.31E+05 | 6.50E+07 | 5.21E+05 | 6.48E+07 | 6.32E+05 |
| 400 | 1.08E+08 | 1.08E+08 | 6.27E+05 | 1.08E+08 | 6.27E+05 | 1.07E+08 | 1.05E+06 | 1.07E+08 | 7.73E+05 | 1.07E+08 | 1.17E+06 |
| 450 | 1.69E+08 | 1.69E+08 | 9.04E+05 | 1.68E+08 | 9.04E+05 | 1.68E+08 | 1.46E+06 | 1.68E+08 | 1.09E+06 | 1.66E+08 | 2.61E+06 |
| 500 | 2.52E+08 | 2.52E+08 | 1.25E+06 | 2.51E+08 | 1.25E+06 | 2.50E+08 | 2.20E+06 | 2.51E+08 | 1.49E+06 | 2.49E+08 | 3.82E+06 |
| 550 | 3.64E+08 | 3.63E+08 | 1.79E+06 | 3.62E+08 | 1.79E+06 | 3.61E+08 | 3.00E+06 | 3.61E+08 | 2.22E+06 | 3.58E+08 | 5.37E+06 |
| 600 | 5.08E+08 | 5.07E+08 | 2.06E+06 | 5.06E+08 | 2.06E+06 | 5.04E+08 | 3.44E+06 | 5.04E+08 | 3.65E+06 | 5.01E+08 | 6.64E+06 |
| 650 | 6.91E+08 | 6.90E+08 | 2.37E+06 | 6.89E+08 | 2.37E+06 | 6.87E+08 | 4.48E+06 | 6.87E+08 | 4.33E+06 | 6.83E+08 | 8.16E+06 |
| 700 | 9.20E+08 | 9.18E+08 | 3.19E+06 | 9.17E+08 | 3.19E+06 | 9.14E+08 | 5.81E+06 | 9.14E+08 | 6.08E+06 | 9.11E+08 | 9.11E+06 |
| 750 | 1.20E+09 | 1.20E+09 | 4.12E+06 | 1.20E+09 | 4.12E+06 | 1.19E+09 | 6.47E+06 | 1.19E+09 | 7.45E+06 | 1.19E+09 | 1.02E+07 |
| 800 | 1.54E+09 | 1.54E+09 | 5.19E+06 | 1.54E+09 | 5.19E+06 | 1.53E+09 | 8.16E+06 | 1.53E+09 | 1.03E+07 | 1.53E+09 | 1.24E+07 |

**Tab. 2.** Number of operations of LUF for different dimensions and densities.

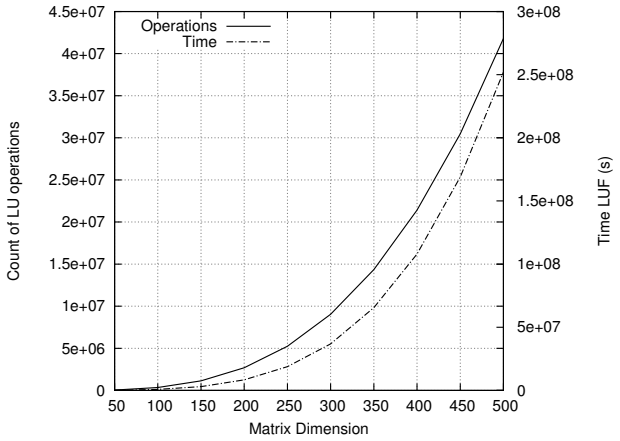


**Fig. 1.** Base count of LU operations vs matrix dimension.

Therefore, we decided to add the algorithm improving the solution of nonlinear equations with one iteration of Newton algorithm for computing of eigenvalues of linear system. Even it was originally developed as iterative algorithm for inversion matrix it can be applied on system of linear equations in very efficient way.

# 6. Matrix-Inversion Iterative Methods

Iterative methods refer to techniques that use successive approximations to obtain a more accurate solution of the linear system at each step. There are so-called stationary methods and the non-stationary methods based on the idea of sequences of orthogonal vectors. Stationary methods perform in each step a same operation on the current values. In the non-stationary method, iterations depend on coefficients. From stationary methods, it can be point out following ones:

- The Jacobi Method (JACOBI)
- The Gauss-Seidel Method (GS)
- The Successive Overrelaxation Method (SOR)

From the non-stationary methods (sometimes referred as Krylov subspace methods), it could be pointed out the following methods:

- Conjugate Gradient Method (CG)
- Generalized Minimal Residual (GMRES)
- Conjugate Gradient Squared Method (CGS)
- BiConjugate Gradient Stabilized (BICGStab)
- BiConjugate Gradient (BICG)
- Quasi-Minimal Residual (QMR)

The original Newton iteration algorithm [43–45] was enhanced for implementation into the core of an electronic circuit simulator. Denoting Jacobian matrix as $\mathbf{J}$ and identity matrix as $\mathbf{I}$, for inverted Jacobian matrix $\mathbf{J}^{-1}$ we can write

$$\mathbf{J}^{-1} = \mathbf{J}^{-1}(2\mathbf{I} - \mathbf{I}) = \mathbf{J}^{-1}(2\mathbf{I} - \mathbf{J}\mathbf{J}^{-1}). \tag{6}$$

For an arbitrary real matrix, $\mathbf{J}$, the above statement allows for each iteration step $n$ to define a generalized inverse (pseudo inverse) denoted as $\mathbf{X}_n$

$$\mathbf{X}_n = \mathbf{J}^{-1} + \boldsymbol{\varepsilon}_n \Rightarrow \mathbf{X}_{n+1} = \mathbf{J}^{-1} + \boldsymbol{\varepsilon}_{n+1} \tag{7}$$

where $\boldsymbol{\varepsilon}_n$ is an error matrix. From that we get to

$$\mathbf{J}^{-1} + \boldsymbol{\varepsilon}_{n+1} = (\mathbf{J}^{-1} + \boldsymbol{\varepsilon}_n)(2\mathbf{I} - \mathbf{J}(\mathbf{J}^{-1} + \boldsymbol{\varepsilon}_n)). \tag{8}$$

The above form can be rewritten using $\mathbf{X}_n$ and $\mathbf{X}_{n+1}$ to a more readable form

$$\mathbf{X}_{n+1} = \mathbf{X}_n\,(2\mathbf{I} - \mathbf{J}\mathbf{X}_n)\,. \tag{9}$$

It is obvious that previous equation will iterate to the solution if

$$0 < \|2\mathbf{I} - \mathbf{J}\mathbf{X}_n\| < 1. \tag{10}$$

It is correct but very impractical for a real implementation. To become a full replacement for standard LUF procedure we need to define an algorithm for obtaining appropriate

initial starting values otherwise the iteration will not converge to a solution.

### 6.1 Starting Values

The algorithm uses results from the LUF method as a starting value. The conversion of this result to higher precision numbers is relatively fast operation involving one matrix vector product. Alternatively, the algorithm can recompute solution with own guess based on (11). This relatively slow procedure can be useful in a case of problematic errors caused by floating point arithmetics.

The result considered here is a special case of a more general and well known Newton iteration method which concerns the iterative computation of pseudo-inverses. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a non-singular matrix and define the sequence $\{\mathbf{X}_n\}_{n\geqq 0}$ of matrices as follows:

$$\begin{cases} \mathbf{X}_0 = \alpha \mathbf{J}^{\mathrm{T}}, & \alpha \in \left(0, \frac{2}{e_0(\mathbf{J}\mathbf{J}^{\mathrm{T}})}\right), \\ \mathbf{X}_{n+1} = \mathbf{X}_n\,(2\mathbf{I} - \mathbf{J}\mathbf{X}_n) \end{cases} \tag{11}$$

where $e_0$ stands for the highest value of the eigenvalue vector $\mathbf{e}$ of the matrix $\mathbf{J}$. Then, $\mathbf{X}_n \to \mathbf{J}^{-1}$ as $n \to \infty$.

This mathematically rigorous definition, unfortunately, requires evaluating eigenvalues. It is not a simple task, and it is very impractical especially in the situation when we want to avoid any factorization method. As a possible solution proved to be reducing the entire definition can be used

$$\mathbf{X}_0 = \beta \mathbf{J}^{\mathrm{T}} \tag{12}$$

where $\beta$ simply belongs to interval $(0, 1)$. If some iteration does not converge, then simulation restarts with a smaller $\beta$ value. It turned out that best value for $\beta$ is somewhere in the interval $(10^{-8}, 10^{-5})$ (for double floating point precision on 64-bit operating system).

### 6.2 Residual and Stopping Criteria

After each iteration, the algorithm needs to evaluate residual to review convergence. The easiest way to do it is to check whether the vector norms of the current and previous solutions show a decreasing trend, i.e.,

$$\|f(\mathbf{X}_{n+1})\| \leqq \|f(\mathbf{X}_n)\| \tag{13}$$

where

$$f(\mathbf{X}_n) = 2\mathbf{I} - \mathbf{J}\mathbf{X}_n. \tag{14}$$

The stopping criteria should be defined by at least two different mechanisms; the first by setting a maximum number of iteration loops to protect from an infinite loop (usually maximum 30 iterations), and also by checking a change between the two consecutive iterations steps. It can be mathematically defined in the following way:

$$\|f(\mathbf{X}_{n+1})\| - \|f(\mathbf{X}_n)\| \leqq \epsilon \tag{15}$$

where $\epsilon$ is a measure of iteration step size, and depending on required precision should be set to at least $10^{-6}$.

## 7. Iterative Method for Linear System

Sticking to the previous notation, a system of linear equations written as a matrix equation can be denoted as

$$\mathbf{A}\mathbf{x} = \mathbf{y} \to \mathbf{x} = \mathbf{A}^{-1}\mathbf{y} \tag{16}$$

where $\mathbf{A}$ is (in general case) an $m \times n$ matrix, $\mathbf{A}^{-1}$ is a symbol of matrix inverse and $\mathbf{x}$ and $\mathbf{y}$ are column vectors with $m$ and $n$ entries, respectively. As we need to avoid factorization methods, again, some iterative algorithm must be used. The simplest solution is to implement well known Gauss-Seidel method [46] for solution of the presented linear system

$$x_i = \frac{c_i - \sum_{j=i}^{n} a_{ij}x_j}{a_{ii}},\ i \neq j \tag{17}$$

where stopping criteria can be defined similarly to Newton iteration algorithm as

$$\|\mathbf{x} - \mathbf{x}_0\| < \epsilon. \tag{18}$$

Usage of the whole algorithm would be impractical and (very) time demanding. Each iteration adds to the simulation additional number of matrix multiplications. Therefore, we use the algorithm only in one iteration as precision booster of the simulation. In other words, the maximum number of iteration loops is limited to one.

## 8. Modified Simulation Algorithm

Putting it all together, we finish with a new procedure that can be implemented to the core of the electronic circuit simulator. The procedure comes with several changes. Apart from special cases of steady state analysis of circuits with oscillators, the simulation process starts with DC analysis. The DC analysis is newly performed by factorization method accompanied with iteration algorithm for the solution of the linear system. In our case, it was simple Newton iteration method. When the transient analysis is the next step in the simulation then results from DC analysis are used as initial starting values for the initial guess of Newton-Raphson algorithm. There is LUF method accompanied with residual computation and our modified Newton iteration method. Full redefined algorithm is written in Algorithm 2.

The core modification of standard algorithm can be seen in initial Jacobian matrix estimation. If the precision of the result is decreased, using NIM (Newton iteration method, see also [47]) as a successive step after the factorization method can significantly increase the accuracy of the result during one iteration loop.

**Algorithm 2.** Modified Simulation Procedure

```
for TIMELINE do
  INITIAL ENUMERATION
  LINEAR SYSTEM
  NIM (precision boost)
  repeat {NONLINEAR SYSTEM (Newton-Raphson)}
    JACOBIAN MATRIX INITIAL EST.
    if not (MEET THE CONDITIONS) then
      DECREASE BETA
      GOTO: JACOBIAN MATRIX INITIAL EST.
    end if
    repeat {JACOBIAN MATRIX}
      NEXT ESTIMATION (vector-matrix product)
      RESIDUAL (vector norm)
    until STOPPING CRITERIA
    NIM (precision boost)
    NEXT ESTIMATION (vector-matrix product)
    RESIDUAL (vector norm)
  until STOPPING CRITERIA
  if not (CONVERGENCE) then
    return CONVERGENCE PROBLEM
  end if
end for
```

## 9. Blockwise Inversion

As it has been shown, the inversion of the big sparse matrices with NIM may be inefficient. To reduce size of the problem we can use blockwise inversion algorithm. It is defined as

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\\ -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\end{bmatrix} \tag{19}$$

where **A**, **B**, **C** and **D** are matrix sub-blocks of arbitrary size. (**A** must be square, so that it can be inverted.) This strategy is particularly advantageous if **A** is diagonal and $\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ is a small. Special property of the electronic circuit matrices allows to reorder matrix in a such way. In that case, inversion of **A** is trivial, and application of NIM algorithm on a relatively small problem with matrices system $\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ increase the total performance of computation.

## 10. Implementing Functional Language

Full implementing of the NR algorithm with the NIM method implemented in the Lisp functional language follows. It was not optimized from the computation performance point of view and should be taken as a demonstration of the first approach to implementation of the simulation procedure. Also, it is obvious from the comparison in Tab. 1 that pure implementation of all operations in Lisp will significantly affect the computation performance. Therefore, it is recommended to precompute the result with a direct method and then in a case of the decreased precision to recompute the results with one iteration loop of the NIM method.

**Listing 1.** Factorisation Free Algorithm in Lisp

```
(let* (
    (residual (eval-function-list function-list))
    (identity-matrix
      (make-identity-list (length *vars*))))
  (loop for nr-iter from 0
    while (and
      (< nr-iter max-nr-iter)
      (> (abs (euclidian-norm residual))
         *epsilon* ))
    do
    (let* (
        (jacobian-matrix
          (eval-jacobian-function-list
            jacobian-function-list))
        (jacobian-inversion-matrix
          (mat-tim (transpose jacobian-matrix)
            *precision*)))
      (loop for ni-iter from 0
        while (and
          (< ni-iter max-ni-iter)
          (>
            (jacobian-residual
              jacobian-matrix
              jacobian-inversion-matrix
              identity-matrix)
            *precision*))
        do
        (setf jacobian-inversion-matrix
          (mult
            jacobian-inversion-matrix
            (matrix-minus
              (mat-tim identity-matrix 2)
              (mult
                jacobian-matrix
                jacobian-inversion-matrix)))))
      (setf-vars *vars*
        (rotate
          (var-matrix-minus (rotate *vars*)
            (mult
              jacobian-inversion-matrix
              (rotate
                (eval-function-list
                  function-list ))))))
      (setf residual
        (eval-function-list
          function-list )))))
```

Implementation of some non-standard functions of the factorization free algorithm may not be obvious. To give the reader a brief description, we include the description of the missing ones. "make-identity-list" will create identity matrix in a form of sparse matrix-vector. "euclidian-norm" will compute Euclidian norm for a given vector. "eval-jacobian-function-list" will evaluate Jacobian sparse matrix, represented by the chained functionals. "transpose" will reindex sparse matrix container according to the matrix transposition. "jacobian-residual" will compute residual for given Jacobian matrix. "mult" is a multiplication of two matrices. "rotate" will reindex sparse vector container according to vector trans-

| Name | Resistor Mesh | CMOS DS Reg. | Big Nonlin. Circ. | Differ. Ampl. | JFET Nonlin. Circ. | CMOS Adder |
|---|---|---|---|---|---|---|
| NIM | 5.760281e-12 | 1.017395e-14 | 8.445384e-12 | 1.333657e-14 | 2.780350e-17 | 1.893448e-15 |
| Jacobi | 8.411095e-01 | 3.588980e-08 | 1.367218e-12 | x | 1.068927e-04 | 5.441340e-18 |
| SOR | 7.675048e-01 | 2.205560e-15 | 8.197609e-05 | x | 8.695059e-05 | 6.617365e-17 |
| GS | 6.948252e-01 | 3.594439e-14 | 1.712990e-12 | x | 8.073145e-05 | 5.986490e-17 |
| CG | 2.288703e-11 | x | x | x | 5.702876e-17 | 6.138830e-11 |
| PCG | 1.973377e-11 | x | x | x | 5.236681e-20 | 1.606757e-11 |
| PCR | 9.439810e-01 | 4.975587e-01 | x | 2.589355e-02 | 1.354612e-12 | 1.535535e-03 |
| GMRES | 1.836753e-12 | 2.159811e-15 | 2.786059e-10 | 1.771027e-11 | 3.363094e-16 | 5.091719e-16 |
| CGS | x | 7.261480e-14 | 1.688600e-11 | 1.629902e-12 | 1.970704e-16 | 5.326264e-15 |
| BICGS | 3.102960e-10 | 3.155781e-14 | 4.999930e-12 | 3.315573e-11 | 3.964052e-18 | 3.197256e-15 |
| QMR | 1.535572e-11 | 3.011353e-14 | 2.292361e-11 | 5.863678e-12 | 2.925510e-17 | 9.527075e-15 |
| Bicg2 | 2.288703e-11 | 2.776918e-14 | 3.008449e-11 | 1.837507e-12 | 5.702876e-17 | 6.762614e-15 |
| Original | 2.550404e-12 | 3.245608e-06 | 1.155214e-12 | 9.103135e-16 | 2.781650e-17 | 7.116835e-05 |

**Tab. 3.** Comparison of different circuit accuracy vs different iterative methods.

position. "matrix-minus" will subtract two sparse matrices of the same size. "mat-tim" is optimized form of multiplication of the matrix and constant number. "eval-function-list" is core function that will evaluate all chained functionals to known values.

# 11. Selected Testing Simulated Circuits

We have selected six electronic circuits of various technology types for comparing the accuracy of the proposed methods (Tab. 3):

- Resistor Mesh – a simple linear circuit interconnected to a large mesh (Fig. 2)
- Big Nonlinear Circuit – a large LED matrix control circuit (Fig. 3)
- A variant of the standard nonlinear circuit with JFET as shown, e.g., in [48]
- Differential Amplifier – a differential amplifier shown in Fig. 4
- CMOS Adder – adder constructed using MOS transistors, this is a standard SPICE (PSpice) demonstration example (called "ADDER - 4 BIT ALL-NAND-GATE BINARY ADDER")
- CMOS DS Regulator – a CMOS stepping regulator circuit as defined in [49]

The results of all the simulations performed with different numerical iterative methods are shown in Tab. 3.

# 12. Results

In Fig. 1, the growth of the number of required operations (addition, multiplication, division) is shown vs increasing size of matrix dimension (full-line and left y-axis). As an informative comparison, it is shown together with required computation time for LU factorization (dot-dashed line, right y-axis). It is important to state that this implementation was programmed as an algorithm using mathematical simulation program Matlab with absolutely no optimization, and its purpose here is to be a reference. Therefore, the presented times here should be taken only as a comparison of a scale. Using appropriate real-world implementation, for example in C language, would be certainly a better choice, but the primary intention was to use unified environment.

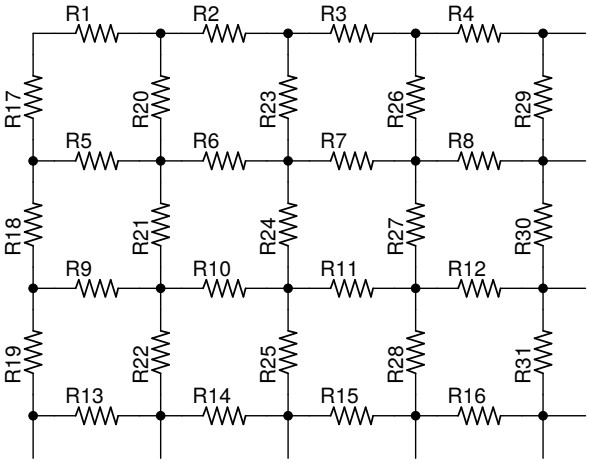


**Fig. 2.** A fragment of the resistor mesh used as a large exemplary linear circuit.

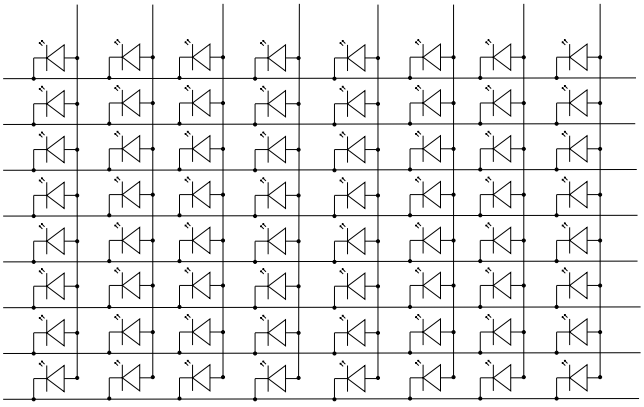

**Fig. 3.** A fragment of the LED matrix control circuit used as a large exemplary nonlinear circuit.

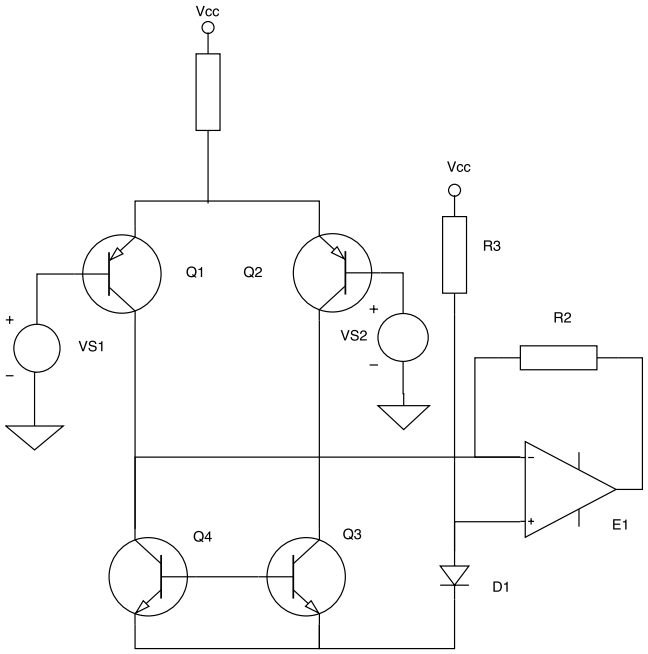


**Fig. 4.** A differential amplifier used as an accuracy test.

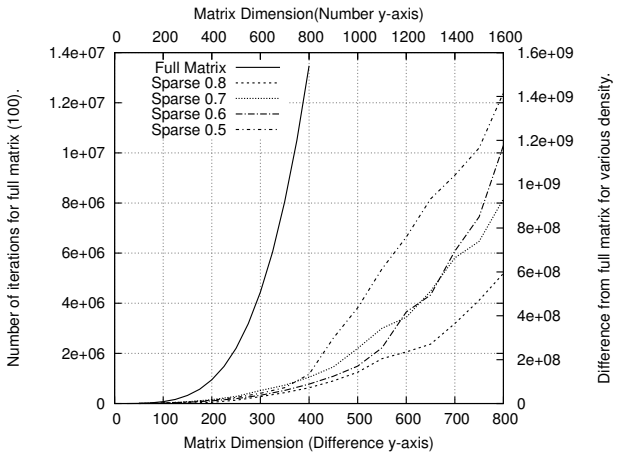


**Fig. 5.** Complex count of LU operations vs matrix dimension.

In Fig. 5, we present a dependency between the count of the LU operations (full line) and matrix dimension. It should be noted that it uses top x-axis, it goes from 0 to 800 and together with the left y-axis. This figure additionally shows, using bottom x-axis and right y-axis, the difference in the number of required iterations of the LU factorization for the matrices with different densities (dot and dash lines). In this case, a sparse matrix system was filled with random numbers. They were generated to have several different non-zero densities from full matrix density 100% to 50% density of nonzero values. The numbers on the right y-axis represent how much fewer operations where needed.

Figure 6 shows final accuracy received for three different circuits of very different circuit sizes (from the point of view of number of nonlinear device models).

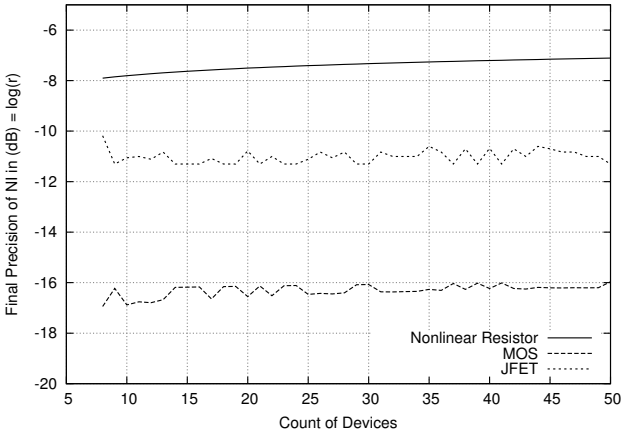


**Fig. 6.** Comparison of achieved accuracy for different circuits sizes.

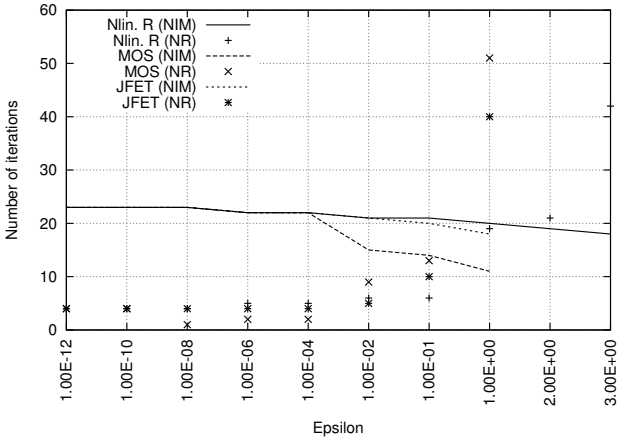


**Fig. 7.** Comparison of minimal accuracy of NIM vs number of iterations.

Used models in this simulation have been: nonlinear resistor defined as pure nonlinear voltage driven resistor, MOS transistor defined by (1), and JFET transistor described by a simplified model derived from original SPICE model [33]. It is presented to demonstrate that when an iterative algorithm is used the precision of the result is limited. It is limited by stopping criteria as well as by floating point precision. Regardless of the required accuracy of the result or a maximum number of iteration loops the accuracy of the results is given primarily by nature of simulated nonlinear equations (device models) and floating point implementation.

Figure 7 demonstrates interaction and influence of precision of inversion matrix on a result of NR method. The x-axis contains different settings of the accuracy of inversion matrix computed by NIM. The y-axis is the number of iterations for different circuits. Each circuit was holding 100 devices of given type. The full-line curve stands for NIM algorithm (evaluation of inversion matrix). Dotted lines represent the number of iterations needed by NR algorithm to find a solution fulfilling stopping criteria. It is nothing

new that the number of iteration loops of NIM depends on a required accuracy of stopping criteria. An interesting observation is that stopping criterion of NIM does not need to be as strict as in NR, and therefore, NIM algorithm can be stopped much earlier which save some computation time. Secondly, there is some point where stopping criteria of NIM algorithm cause very steep growth of the required number of iteration loops on NR algorithm or even its divergence.

Although it proves to be very stable method, it is clearly visible now that the real implementation suffers from one big drawback apart from other iterative algorithms and of course from direct LUF. It is a need of computing and storing of at least one additional (unfortunately dense) matrix. Simplicity of the algorithm allows compiler to optimize the code in such a way that performance is to be rapidly increased, but with increase of matrix dimension memory, requirements of this method rapidly decrease performance. This can be visible from Fig. 8. It uses the following naming conventions:

- (LUF BFS n.) LU factorization with forward elimination and back substitution
- (NIM n. -Ofast) Naive implementation of algorithm
- (LUF BFS naive -Ofast) Compiled with full support of compiler optimization
- (LUF BFS Lapack) Algorithms were implemented using Lapack functions
- (NIM Lapack) Newton Iteration method
- (GNU Octave) Algorithm for matrix inversion implemented in GNU Octave
- (Cholesky Lapack) Computation uses Cholesky method

It can be seen from the results that "naive" implementation that represents (only) implementation of the algorithm without respect to procesor cache and fast computation techniques slows entire computation by several orders. Number of required iterations to obtain the result with particular precision unfortunately depends on the initial estimation and matrix dimension.

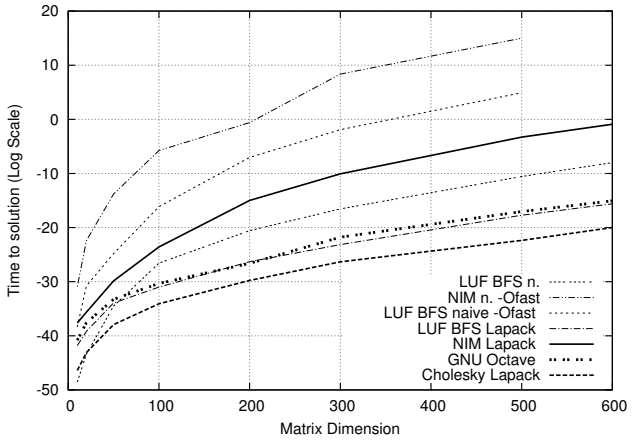


**Fig. 8.** Comparison of performance of method implementations.

## 12.1 Comparing Tables of Method Accuracies

In Tab. 4, there is a comparison of the accuracy increase with additional NIM cycle computed with arbitrary numbers to matrix dimension. It got an additional accuracy boost computed by NIM after each successful iteration time step. The adapted transient analysis completed by single loop NIM with arbitrary precision is shown in Algorithm 2. It can be applied to the end of operating-point analysis or as a final procedure after each time step of transient analysis.

The values in Tabs. 3 and 4 represent maximal accuracy of the computation of RHS vector that could be achieved with given method on the specific simulation task. The measure of accuracy is limited by mathematical properties of given numerical method from one side and by digital representation of floating point numbers from the other side. While Tab. 3 gives an idea on rate of change of the accuracy levels, Tab. 4 demonstrates possible improvement in accuracy given by proposed method. Therefore, the values in the tables do *not* represent the results of the simulations, but the inaccuracy of the calculation from the ideal state.

| Circuit | Accuracy | Accuracy Boost |
|---|---|---|
| Resistor Mesh | 1.37E-10 | 2.20E-19 |
| Big Nonlin. Circ. | 6.32E-15 | 1.89E-29 |
| JFET Nonlin. Circ. | 2.14E-15 | 5.21E-29 |
| Differ. Ampl. | 4.03E-13 | 2.69E-27 |
| CMOS Adder | 1.16E-11 | 1.27E-24 |
| CMOS DS Reg. | 8.96E-14 | 5.47E-28 |

**Tab. 4.** Accuracy boost given by one iteration loop of NIM with arbitrary precision.

# 13. Conclusion

Basing on the assumption that electronic device models definitions are introduced to the simulation as chained functionals, they will autonomously evaluate themselves to values upon call. It is evident that trying to convert those entries to matrix system to compute matrix inversion or solution of the linear system is redundant operation. It seems to be as a solution to chain all entries to another function that on call evaluate entire simulation. The key factor of the implementation lies in the use of the iterative recursion. In this article, we presented the new procedure utilizing factorization free algorithm that can enhance standard computation core of electronic circuit simulators.

Not only from the results it is evident that accuracy or better convergence of iterative numerical methods depends on matrix properties and also on starting values. It means that in the case when LUF method does finish with sufficient precision then better results can be achieved with numerical iteration (with an assumption that same numeric floating types are used in both scenarios). The article suggests that accuracy of the results should be presented alongside with results and of course in a situation when the standard method does not achieve reasonable accuracy then some iteration

method can be used next. The concrete method depends on the problem and picks up could be a tricky part. We suggest NIM approach as very universal solution, but it is of course very memory demanding and can be very inefficient for huge matrixes.

As it has been already said, NIM method supports a use of rationals (numbers expressed as the fraction $p/q$ where $p$ and $q$ are integers and $q$ is nonequal to zero). When iteration utilizes computation with rationals, it is possible with NIM to receive a more precise solution. Performing standard direct method with rational numbers has significant drawbacks that are for wider discussion. Mainly, standard operations with rationals are slower than those with standard types. A large difference has been presented between evaluating of one million operations in C language and Lisp with a use of rationals or with floating point numbers. The significant problem that arises with rational numbers is a size of the numeric container and required time to store and obtain it from computer memory. Our tests showed that it takes 0.7 s to store the rational number of size $10^{1000000}$. Usage of one iteration of NIM with initial starting value as a result from previous factorization allows to use arbitrary numbers in much more effective way and, therefore, obtain the final solution with improved precision faster.

All computation and simulation tests were made on the same computer machine with a processor Intel Core i3 4160 Haswell, chipset Intel H87 and RAM 8GB DDR3, with 64b architecture and operating system.

## Acknowledgments

This paper was supported by the Technology Agency of the Czech Republic under the grant No. TE01020186, and by the Grant Agency of the Czech Technical University in Prague under the grant No. SGS18/079/OHK3/1T/13.

## References

[1] SAIZ-VELA, A., MIRIBEL-CATALA, P., COLOMER, J., et al. Accurate design of high-voltage multistage voltage doublers based on compact mathematical model. *Electronics Letters*, 2007, vol. 43, no. 15, p. 797–798. ISSN: 0013-5194. DOI: 10.1049/el:20070405

[2] SANYAL, A., RASTOGI, A., CHEN, W., et al. An efficient technique for leakage current estimation in nanoscaled CMOS circuits incorporating self-loading effects. *IEEE Transactions on Computers*, 2010, vol. 59, no. 7, p. 922–932. ISSN: 0018-9340. DOI: 10.1109/TC.2010.75

[3] YANG, S., LIU, S., FENG, W., et al. SPICE circuit model of voltage excitation fluxgate sensor. *IET Science, Measurement Technology*, 2013, vol. 7, no. 3, p. 145–150. ISSN: 1751-8822. DOI: 10.1049/iet-smt.2013.0005

[4] KUMAWAT, R., SAHULA, V., GAUR, M. Probabilistic model for nanocell reliability evaluation in presence of transient errors. *IET Computers Digital Techniques*, 2015, vol. 9, no. 4, p. 213–220. ISSN: 1751-8601. DOI: 10.1049/iet-cdt.2014.0124

[5] VAN UFFELEN, M., GEBOERS, S., LEROUX, P., et al. SPICE modelling of a discrete COTS SiGe HBT for digital applications up to MGy dose levels. *IEEE Transactions on Nuclear Science*, 2006, vol. 53, no. 4, p. 1945–1949. ISSN: 0018-9499. DOI: 10.1109/TNS.2006.880949

[6] HUSZKA, Z., CHAKRAVORTY, A. Implementation of delay-time-based nonquasi-static bipolar transistor models in circuit simulators. *IEEE Transactions on Electron Devices*, 2014, vol. 61, no. 8, p. 3004–3006. ISSN: 0018-9383. DOI: 10.1109/TED.2014.2327664

[7] TANAKA, C., SAITOH, M., OTA, K., et al. SPICE-Based performance analysis of trigate silicon nanowire CMOS circuits. *IEEE Transactions on Electron Devices*, 2013, vol. 60, no. 4, p. 1451–1456. ISSN: 0018-9383. DOI: 10.1109/TED.2013.2247607

[8] STEINER, M., SIEFER, G., BETT, A. SPICE network simulation to calculate thermal runaway in III-V solar cells in CPV modules. *IEEE Journal of Photovoltaics*, 2014, vol. 4, no. 2, p. 749–754. ISSN: 2156-3381. DOI: 10.1109/JPHOTOV.2014.2299398

[9] LIN, J., TOH, E. H., SHEN, C., et al. Compact HSPICE model for IMOS device. *Electronics Letters*, 2008, vol. 44, no. 2, p. 91–92. ISSN: 0013-5194. DOI: 10.1049/el:20083116

[10] WONG, O. Y., WONG, H., TAM, W. S., et al. Dynamic analysis of two-phase switched-capacitor DC-DC converters. *IEEE Transactions on Power Electronics*, 2014, vol. 29, no. 1, p. 302–317. ISSN: 0885-8993. DOI: 10.1109/TPEL.2013.2249594

[11] KE, H., HUBING, T., MARADEI, F. Using the LU recombination method to extend the application of circuit-oriented finite element methods to arbitrarily low frequencies. *IEEE Transactions on Microwave Theory and Techniques*, 2010, vol. 58, no. 5, p. 1189–1195. ISSN: 0018-9480. DOI: 10.1109/TMTT.2010.2045533

[12] ACARY, V., BONNEFON, O., BROGLIATO, B. Time-stepping numerical simulation of switched circuits within the nonsmooth dynamical systems approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2010, vol. 29, no. 7, p. 1042–1055. ISSN: 0278-0070. DOI: 10.1109/TCAD.2010.2049134

[13] FERREIRA, D., OLIVEIRA, J., PEDRO, J. A novel time-domain CAD technique based on automatic time-slot division for the numerical simulation of highly nonlinear RF circuits. *IEEE Transactions on Microwave Theory and Techniques*, 2014, vol. 62, no. 1, p. 18–27. ISSN: 0018-9480. DOI: 10.1109/TMTT.2013.2293481

[14] ZHANG, X., CHEN, W. H., FENG, Z. Novel SPICE compatible partial-element equivalent-circuit model for 3-D structures. *IEEE Transactions on Microwave Theory and Techniques*, 2009, vol. 57, no. 11, p. 2808–2815. ISSN: 0018-9480. DOI: 10.1109/TMTT.2009.2032462

[15] SAFAVI, S., EKMAN J. A hybrid PEEC-SPICE method for time-domain simulation of mixed nonlinear circuits and electromagnetic problems. *IEEE Transactions on Electromagnetic Compatibility*, 2014, vol. 56, no. 4, p. 912–922. ISSN: 0018-9375. DOI: 10.1109/TEMC.2014.2300372

[16] FRANCO, F., PALOMAR, C., IZQUIERDO, J., et al. SPICE simulations of single event transients in bipolar analog integrated circuits using public information and free open source tools. *IEEE Transactions on Nuclear Science*, 2015, vol. 62, no. 4, p. 1625–1633. ISSN: 0018-9499. DOI: 10.1109/TNS.2015.2416000

[17] TLELO CUAUTLE, E., RODRIGUEZ CHAVEZ, S. Graph-based symbolic technique for improving sensitivity analysis in analog integrated circuits. *IEEE Latin America Transactions*, 2014, vol. 12, no. 5, p. 871–876. ISSN: 1548-0992. DOI: 10.1109/TLA.2014.6872898

[18] KAPRE, N., DEHON, A. SPICE2 : Spatial processors interconnected for concurrent execution for accelerating the SPICE circuit simulator using an FPGA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012, vol. 31, no. 1, p. 9–22. ISSN: 0278-0070. DOI: 10.1109/TCAD.2011.2173199

[19] CHEN, X., REN, L., WANG, Y., et al. GPU-Accelerated sparse LU factorization for circuit simulation with performance modeling. *IEEE Transactions on Parallel and Distributed Systems*, 2015, vol. 26, no. 3, p. 786–795. ISSN: 1045-9219. DOI: 10.1109/TPDS.2014.2312199

[20] ZHOU, T., LIU, H., ZHOU, D., et al. A fast analog circuit analysis algorithm for design modification and verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2011, vol. 30, no. 2, p. 308–313. ISSN: 0278-0070. DOI: 10.1109/TCAD.2010.2081750

[21] FAKHFAKH, M., TLELO-CUAUTLE, E., FERNÁNDEZ, F. V. *Design of Analog Circuits through Symbolic Analysis*. Bentham Science Publishers, 2012. ISBN: 9781608054251

[22] FAKHFAKH, M., TLELO-CUAUTLE, E., CASTRO-LOPEZ, R. *Analog/RF and Mixed-Signal Circuit Systematic Design*. Berlin Heidelberg: Springer-Verlag, 2013. ISBN: 9783642363283

[23] DOBEŠ, J., MÍCHAL, J., BIOLKOVÁ, V. Multiobjective optimization for electronic circuit design in time and frequency domains. *Radioengineering*, 2013, vol. 22, no. 1, p. 136–152. ISSN: 1210-2512

[24] STEFAŃSKI, T. P. Electromagnetic problems requiring high-precision computation. *IEEE Antennas and Propagation Magazine*, 2013, vol. 55, no. 2, p. 344–353. ISSN: 1045-9243. DOI: 10.1109/MAP.2013.6529388

[25] DOBEŠ, J., ČERNÝ, D., VEJRAŽKA, F., et al. Comparing the steady-state procedures based on epsilon-algorithm and sensitivity analysis. In *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. Cairo (Egypt), 2015, p. 1–4. DOI: 10.1109/ICECS.2015.7440388

[26] FAKHFAKH, M., TLELO-CUAUTLE, E., SIARRY, P. *Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design*. Cham: Springer, 2015. ISBN: 9783319198712

[27] ČERNÝ, D., DOBEŠ, J. An efficient procedure for transient analysis of electronic circuits with increased precision. *MATEC Web of Conferences*, EDP Sciences, 2016, vol. 76. DOI: 10.1051/matecconf/20167601007

[28] ELRAD, T., FILMAN, R. E., BADER, A. Aspect-oriented programming: Introduction. *Communications of the ACM*, 2001, vol. 44, no. 10, p. 29–32. DOI: 10.1145/383845.383853

[29] CHEN, Y., ACAR, U. A., TANGWONGSAN, K. Functional programming for dynamic and large data with self-adjusting computation. *ACM SIGPLAN Notices*, 2014, vol. 49, no. 9, p. 227–240. DOI: 10.1145/2692915.2628150

[30] ZAHARIA, M., CHOWDHURY, N. M. M., FRANKLIN, M. J., et al. *Spark: Cluster Computing with Working Sets*. Tech. Rep. UCB/EECS-2010-53. Electrical Engineering and Computer Sciences, University of California at Berkeley, 2010.

[31] ACAR, U. A., BLELLOCH, G. E., HARPER, R. Adaptive functional programming. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2006, vol. 28, no. 6, p. 990–1034. DOI: 10.1145/1186632.1186634

[32] MAWBY, P., IGIC, P., TOWERS, M. Physically based compact device models for circuit modelling applications. *Microelectronics Journal*, 2001, vol. 32, no. 5, p. 433–447. DOI: 10.1016/S0026-2692(01)00013-1

[33] ANTOGNETTI, P., MASSOBRIO, G. *Semiconductor Device Modeling with Spice*. New York: McGraw-Hill, Inc., 1993. ISBN: 0070021538

[34] VLADIMIRESCU A. *The Spice Book*. New York: J. Wiley & Sons, 1994. ISBN: 0471609269

[35] ČERNÝ, D., DOBEŠ, J., NAVRÁTIL, V. Functional chaining mechanism allowing definable models of electronic devices. In *Proceedings of the European Conference on Circuit Theory and Design (ECCTD)*. Catania (Italy), 2017. DOI: 10.1109/ECCTD.2017.8093250

[36] JIAN, S. L., LU, K., WANG X. P. A survey on concepts and the state of the art of functional programming languages. In *Systems and Computer Technology: Proceedings of the International Symposium on Systems and Computer technology (ISSCT)*. Shanghai (China), 2014, p. 71–77. ISBN: 9781138028722

[37] ČERNÝ, D., DOBEŠ, J. Common LISP as simulation program (CLASP) of electronic circuits. *Radioengineering*, 2011, vol. 20, no. 4, p. 880–889. ISSN: 1210-2512

[38] ČERNÝ, D., DOBEŠ, J. Functional programming languages in computer simulation of electronics circuits. In *Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI)*. Las Vegas (USA), 2014, vol. 1, p. 229–234. DOI: 10.1109/CSCI.2014.46

[39] QUARLES, T. L. *Analysis of Performance and Convergence Issues for Circuit Simulation*, Doctoral Dissertation, University of California, Electronics Research Lab, Berkeley, 1989.

[40] DAVIS, T. A., YEW, P. C. A nondeterministic parallel algorithm for general unsymmetric sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications*, 1990, vol. 11, no. 3, p. 383–402. DOI: 10.1137/0611028

[41] CHEN, X., WANG, Y., YANG, H. An adaptive LU factorization algorithm for parallel circuit simulation. In *Proceedings of the 17th Asia and South Pacific Design Automation Conference (ASP-DAC)*. Sydney (Australia), 2012, p. 359–364. ISSN: 2153-6961. DOI: 10.1109/ASPDAC.2012.6164974

[42] KAPRE, N., DEHON, A. Parallelizing sparse matrix solve for SPICE circuit simulation using FPGAs. In *Proceedings of the International Conference on Field-Programmable Technology (FPT)*. Sydney (Australia), 2009, p. 190–198. DOI: 10.1109/FPT.2009.5377665

[43] BEN-ISRAEL, A., GREVILLE, T. N. *Generalized Inverses: Theory and Applications*. Springer Science & Business Media, 2003. ISBN: 0387002936

[44] BEN-ISRAEL, A. An iterative method for computing the generalized inverse of an arbitrary matrix. *Mathematics of Computation*, 1965, vol. 19, no. 91, p. 452–455. DOI: 10.2307/2003676

[45] BEN-ISRAEL, A., COHEN, D. On iterative computation of generalized inverses and associated projections. *SIAM Journal on Numerical Analysis*, 1966, vol. 3, no. 3, p. 410–419. DOI: 10.1137/0703035

[46] HAGEMAN, L. A., YOUNG, D. M. *Applied Iterative Methods*. Courier Corporation, 2012. ISBN: 048643477X

[47] KOROVKIN, N. V., CHECHURIN, V. L., HAYAKAWA, M. *Inverse Problems in Electric Circuits and Electromagnetics*. New York: Springer Science+Business Media, 2007. ISBN: 9780387335247

[48] WOLFF, F. G. Spice Amplifier Tutorial. In *Spice3/Bandwidth/Slew Notes*. VLSI CAD Group, EECS department, Case Western Reserve University. [Online]. Available at: http://www2.eng.cam.ac.uk/~dmh/ptialcd/jfet/tut_spice3_jfet_bias.html. Cited 2018-06-21.

[49] SCHWEBER, B. *Understanding the Advantages and Disadvantages of Linear Regulators*. Digi-Key's North American Editors, Sep. 2017. [Online] Cited 2018-06-21. Available at: https://www.digikey.com/en/articles/techzone/2017/sep/understanding-the-advantages-and-disadvantages-of-linear-regulators.

# About the Authors . . .

**David ČERNÝ** was born in Prague, Czech Republic, in 1985. He received his M.Sc. degree in 2009 from the Faculty of Electrical Engineering of the Czech Technical University in Prague. He finished his Ph.D. study at the Department of Radioelectronics at Czech Technical University in Prague in 2017 and received Ph.D. degree. His research interests include simulation of high frequency circuits, physical modeling of electrical devices, and very large-scale integrated circuit analysis.

**Josef DOBEŠ** received the Ph.D. degree in Microelectronics from the Czech Technical University in Prague in 1986. From 1986 to 1992, he was a researcher of TESLA Research Institute. He is currently with the Department of Radioelectronics of the Czech Technical University in Prague. His current research interests include physical modeling of elements of electronic circuits, especially radio-frequency and microwave transistors and transmission lines, creating or improving special algorithms for circuit analysis and optimization such as time- and frequency-domain sensitivity, poles-zeros or steady-state analyses, and creating a CAD tool for analysis and optimization of radio-frequency circuits.

**Stanislav BANÁŠ** was born in 1970. He received his M.Sc. degree in Electrical Engineering from the Technical University in Brno in 1994. Last year of his study he spent in scholarship in CNRS institute in Grenoble, where he was interested in optoelectronic properties of hydrogenated amorphous silicon. From 1996 he works as a modeling and characterization engineer in Motorola Czech Design Center in Roznov, later transferred to ON Semiconductor SCG Czech Design Center in Roznov. From 2012 he studies for Ph.D. in Technical University in Prague. His research interests include the modeling of high-voltage semiconductor components.